(to appear) Proceedings of the Second International Workshop on Requirements Engineering: Foundation for Software Quality, Jyvaskyla, Finland, June 12-13, 1995

Requirements Management and its Support by Process-centered Environments

Ernst Ellmer[(1)], Dieter Merkl[(2)], A Min Tjoa[(2)]

[(1)]Universität Wien, Institut für Informationssysteme
Liebiggasse 4, A-1010 Wien, Austria
email: ee@ifs.univie.ac.at
[(2)]Technische Universität Wien, Institut für Softwaretechnik
Resselgasse 3, A-1040 Wien, Austria
email: {dieter, tjoa}@ifs.tuwien.ac.at

Abstract. *This paper focuses on the management of the requirements engineering process and on the contributions a process supporting environment (PSE) can provide to perform this task. We use the key practices defined by the capability maturity model for software (CMM), one of the most prominent software process improvement approaches, as a requirements management scenario. For each activity proposed by the CMM, we investigate possible support by PSE functionality. For this purpose, we develop a functional PSE reference model based on current research and prototypes. In a last step, we summarize the results of our investigations to obtain some general conclusions on possible PSE support for requirements management processes.*

1. Introduction

The desired output of the requirements engineering process is "*a formally represented complete system specification on which agreement has been achieved*" [36] and thus requirements engineering (RE) is an early phase in the software life cycle. This makes it a challenging task for software engineers and interesting for researchers. Finding or even avoiding faults during requirements engineering guards against problems during design, implementation, maintenance, the other phases in the software life cycle according to the waterfall model [39]. Errors which are detected during RE can be corrected easier and at lower costs than in the later phases. The better requirements engineering is accomplished the higher the probability to reach the ultimate goal of software development, namely to produce high quality software at reasonable costs. This leads our attention to the management of the requirements engineering process and its support by environments.

In the recent years a number of applications were developed under the umbrella of CSCW (Computer Supported Cooperative Work) [18] or groupware [27]. Clarence Ellis et al. define groupware as "*computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment*" [14]. We feel that requirements engineering is a task as characterized in the above definition to be supported by groupware, and thus groupware applications are worth considering when searching for tool support for requirements engineering.

Requirements engineering and groupware are the two building blocks of our paper. Our aim is to investigate how a special kind of CSCW applications, namely process supporting environments (PSEs), may contribute to the management of the requirements engineering process.

1.1 Structure and Outline of our Paper

Our investigations are based on a requirements management scenario and on a PSE reference model. The PSE reference model is developed in section two and is based on current research in the area of process supporting environments. Two communities focus on PSEs.

These are the software process community on the one hand and the workflow community on the other hand. Our reference model represents a unifying view on the two kinds of process supporting environments. Nevertheless, it focuses on process-centered software engineering environments, the systems developed by the software process community, because they are intended to support software development. This focus is motivated by the fact that requirements engineering is a software development activity rather than an office task supported by workflow management systems, the applications of the workflow community. The requirements management scenario is based on the requirements management key process area of the capability maturity model for software (CMM) [23, 34, 35]. The CMM was developed at the Carnegie Mellon Software Engineering Institute (CM/SEI) and is one of the most prominent software process improvement approaches. In section three we briefly review the structure of the CMM and the parts needed for our investigations. In section four we then examine in detail how a PSE according to our reference model is able to support the key practices of the requirements management key process area as defined by the CMM. This section thus builds the core of our paper. Section five summarizes the results of our investigations and section six concludes the paper.
We would like to point out the two basic assumptions our paper relies on and argue why we feel they are reasonable.

- First, we suppose a process supporting environment to provide the functionality of our reference model. This however is an optimistic assumption because currently no system providing the whole range of this functionality exists. Yet, our reference model is realistic in the sense that each function we define is target of current research and provided by different research prototypes. Therefore we provide an extensive list of references. We feel that this look into future is reasonable because it encourages, motivates, and shows the usefulness of future research on PSE support for requirements management.
- Second, we suppose the key practices of the requirements management key process area of the CMM to be a reasonable way to manage the requirements engineering process. We feel several reasons plead for this assumption. The CMM is carefully developed and subject to continuous research and improvement for many years. Its practical usage is shown and proven by several case studies [24, 20, 30]. Beyond that, the clear structure and exact definition of the CMM key process areas and key practices favors our investigations.

2. Characterizing PSEs

The goal of a process supporting environment (PSE) is the support, guidance and (partly) automation of business processes[1]. In order to characterize PSEs, we refer to [8, 10] where a software process is divided into *production process*, *meta process* and *process support*. The production process includes all activities to produce the software product, e.g. drawing an entity relationship diagram during requirements analysis, coding a procedure during implementation, checking a routine during testing. The meta process on the other hand includes all activities to evolve the software process, e.g. modeling the process, instantiating the process, monitoring the process, and so on. The process support provides both, computerized tools to support production process activities, e.g. an editor to draw an entity relationship diagram, and a computerized environment to support meta process activities, e.g. a tool to monitor the actual state of process execution. What a PSE according to our definition has to provide is meta process support on the one hand, and a flexible mechanism to incorporate production process support on the other hand. Our reference model has to be viewed under the light of these definitions.
Further, we distinguish between the view of a user and the view of a developer of a PSE on such an environment. For the first one, the functionality offered by the environment is

1. Please note that the term “business process” in the context of this paper is used in a broader sense than simply “efforts to make money”.

important, while the second one focuses on technologies to develop an environment providing this functionality. Examples of such infrastructure technologies for PSEs are distributed systems, transactions, concurrency control, versioning, security models, databases, and so on. Our reference model is a functional one and thus represents the view of a user on the PSE. Figure 1 gives an overview of our reference model. It consists of data (artifacts) and functions manipulating these data. Data represent the knowledge necessary to manage process performance. Functions access and manipulate the data. We distinguish between two types of functions. These are modeling functions on the one hand and runtime supporting functions on the other hand. The task of the modeling functions is to prepare processes for performance by a PSE, whereas process supporting functions then enable guidance and maybe automatization of process performance. In the rest of this section we will describe the artifacts as well as the functions shown as in Figure 1in more detail.

2.1 PSE Artifacts

A PSE has to manage both, the static structure and the dynamic behavior of a software development organization. By "static structure" we denote the organizational units that take part in a project whereas the term "dynamic behavior" refers to the state of the projects carried out both, at present and in future.

Organizational Model (OM)

An **organizational model OM** represents the organizational structure of a corporation which defines organizational units performing processes. An organizational unit may be a software project team as well as a department or a single individual.

Process Models (PM)

PM is a set of process models representing computerized models of the projects a software development organization has to perform. According to [16] we define a process model as any abstract representation of a process without specifying a certain level of abstraction.

Processes (P)

P is a set of enactable or currently enacting[1] **processes.** Each process of P is an enactable instance of a process model out of PM in a certain state of execution. Be aware of the difference between a process model and an enactable instance of the process modeled. While the first one describes the structure and behavior of one process type, the second one represents a concrete process instance.

2.2 PSE Functions

We distinguish PSE functionality according to the phase of the process life cycle they belong to. First, we identify what we call modeling functionality supporting the whole set of activities carried out to prepare processes for enactment. Second, we identify run-time supporting functionality managing the enactment of processes.

Organizational Modeling (OMod)

Organizational modeling OMod enables a user to develop a model *OM* of a software development organization and thus providing organizational knowledge to the environment. Example organizational modeling approaches are ORM [40] or TOSCA [37, 38].

Process Modeling (PMod)

The activity of **process modeling PMod** is an abstraction process filtering out the information of a real world process relevant for the purposes of the resulting process model. The process modeling process is determined by *process modeling concepts* (PMC) comprising a set of abstractions for process modeling [26], a *process modeling language* (PML) providing (graphical or textual) formalisms for representing process models [2], and a *process modeling methodology* (PMM) defining a way (an algorithm) to generate a process model of a certain process [32]. Process modeling is an area of intensive research. Objectives of

1. Please note that we use the term enaction instead of execution to stress the integration of "human processors" in process performance.

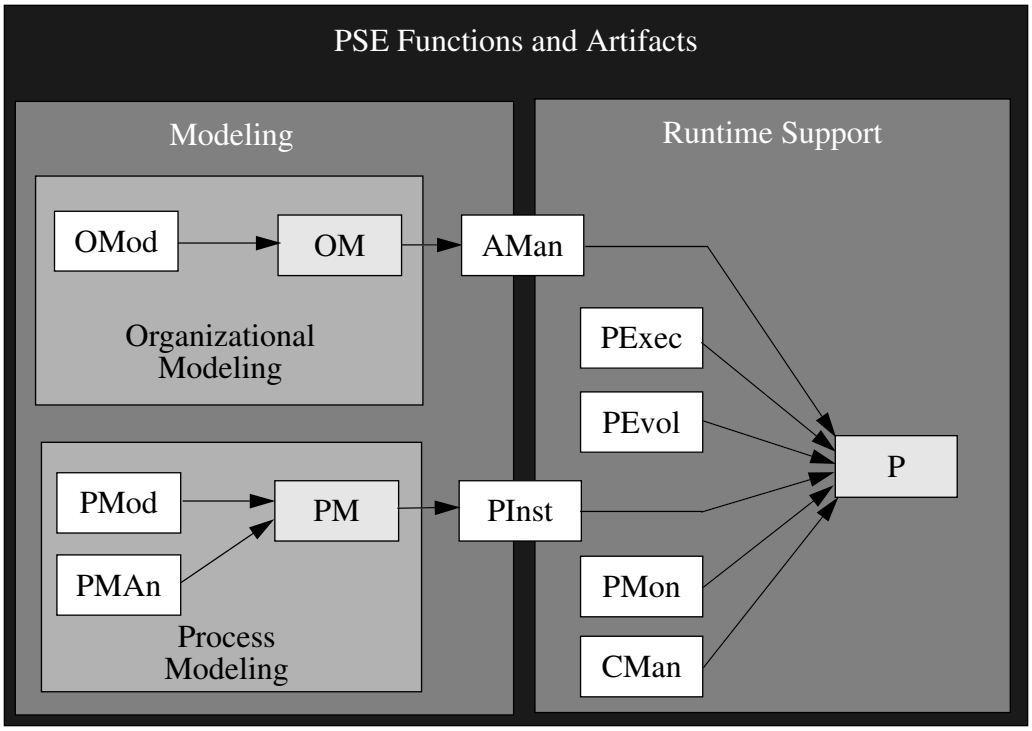

Figure 1: A Functional View on a PSE

process modeling are discussed in [11, 2, 29, 31], existing approaches are reviewed in [17, 2, 11].

Process Model Analysis (PMAn)

Process model analysis PMAn allows to analyze a process model in order to ensure the desired process performance. [19] states that software process model analysis "means to check if the information expressed in the model is sufficient to ensure the software process behavior the modeler had in mind" and identifies validation, verification, and post-evaluation as three branches of process model analysis. Much work on process model analysis was done by the MELMAC team [13].

Process Execution Support (PExec)

Process execution support PExec in our reference model stands for enabling the incorporation of production process activities in a software process supporting environment. Process execution support as defined within our reference model focuses on two services. The first service is user notification referring to an interface that tells the user of a PSE three things, namely when, i.e. at which point in time, to do what, i.e. a subactivities of a process, and how, i.e. a methodology or tool. The second service is referred to as tool integration. This is an interface to the production process supporting tools that enabling a flexible arrangement of the working environment. [41] investigates how to use tools from within a PSE. Process execution support functionality comprises the production process support of a PSE as defined earlier in this paper.

Process Monitoring (PMon)

Process monitoring PMon is needed for monitoring the state of the execution of running processes. Research on process monitoring and process measurement as gathering the data to be monitored is at an initial stage [22]. Efforts in this direction are undertaken by the SPADE team [4], by the ADELE team [Belkathir94], and by the Arcadia team [1].

Process Evolution (PEvol)

Process evolution PEvol enables activities necessary to capture possible evolutions of running processes. Process evolution is an area of intensive research in the PSE research community [9, 21]. Example research prototypes focusing on process evolution are SPADE [3],

EPOS [25], Marvel [28]. [5] provide a conceptual framework for process evolution in PSEs.

Configuration Management (CMan)

Configuration management CMan provides functionality to manage the products of processes, e.g. versions, releases. Configuration management is agreed to be an important part of a SEE. [12, 15] describe the past, present, and future of configuration management and give examples of configuration management models. Example systems addressing configuration management are ADELE [6] and SCALE [33].

Process Instantiation (PInst)

Process instantiation PInst creates an enactable process instance out of a process model.

Agent Management (AMan)

Agent management AMan allows to associate agents to activities of processes in order to perform these processes. Agents are modeled in the organizational model *OM* while activities are part of processes *P* modeled by process models *PM*. Agent management thus defines a mapping between organizational units and process activities. Further agent management allows to generate agent requirements, e.g. project team structure out of process information. While organizational modeling supports the representation of the static, process independent structure of an enterprise, agent management is concerned with organizational settings due to process behavior.

3. The CMM Revisited

This section is based on the description of the CMM V1.1 in [34, 35]. The goals of the CMM are identified as assessing and improving software processes as well as assessing the capabilities of possible contractors.

The structure of the CMM is the following:

- A *maturity level* is "a well-defined evolutionary plateau toward achieving a mature software process" [35]. The CMM consists of five maturity levels which constitute its top level structure. The maturity level of an organizations software process indicates its process capabilities. The five maturity levels are:
  - *Initial level*: The main characteristics of a software process at the initial level are its ad hoc and chaotic nature. No kind of project management is implemented.
  - *Repeatable level*: At the repeatable level basic project management policies are established and implemented. The problem is that this management is only based on experiences with similar projects.
  - *Defined level*: The software process of an organization reaches the defined level if an organization-wide standard process is established and a process group responsible for developing and maintaining the standard process exists.
  - *Managed level*: An organizational measurement program measuring the quality of the software products and processes is established. Measurement data is stored in an organization-wide process database.
  - *Optimizing level*: The focus of the entire organization lies on process improvement. The software process is monitored to be able to react immediately on problems or even anticipate them.
- The maturity levels - except the initial one - are decomposed into *key process areas*. A key process area "identifies the issues that must be addressed to achieve a maturity level" [34]. Performing the activities of the set of key process areas associated to a maturity level means achieving a set of goals important for enhancing process capability.
- The key process areas are organized by *common features* indicating whether implementation and institutionalization of a key process area is effective, repeatable, and lasting. [35] defines the following common features.
  - *Commitment to perform* describes those actions necessary to ensure that the process is established and will endure.

- *Ability to perform* describes preconditions to exist in the project or organization to implement the software process completely.
- *Activities performed* describe the procedures to be performed to implement a key process area.
- *Measurement and analysis* includes measurement and analysis needs of the process.
- *Verifying implementation* comprises steps to ensure activities are carried out in accordance with the defined process.

- The lowest level structural part of the CMM are the *key practices*. The key practices "describe the infrastructure and activities that contribute most to the effective implementation and institutionalization of the key process area" [34].

4. PSE Support for Requirements Management

The requirements engineering process as part of the software process can be divided into meta process and production process as described in section two. The requirements management key process area of the CMM addresses the meta process of requirements engineering while its production process is addressed by the software product engineering key process area. We investigate how a PSE can support requirements management according to the CMM. This means, we elaborate possible PSE support for the requirements engineering meta process. Support of the RE production process by a PSE looks like also described in section two. It is limited to the notification of agents about what (which activities) to do when (at which time) and how (using which methodology, tool), and on the integration of production process supporting tools into the environment.

Before we go into details, we state the two goals of the requirements management key process area. These are first, to control system requirements allocated to software in order to establish a baseline for software engineering and management use, and second, to keep software plans, products, and activities consistent with the requirements allocated to software. The requirements allocated to software are defined in [35] as the customers requirements that will be addressed by the software project.

In the remainder of this section we will examine a typical requirements management scenario for possible PSE support. This examination is based on the key practices of the associated CMM key process areas. Furthermore the key practices are ordered according to the common features and are uniformly structured as follows.

Key Practice: *Description of the key practice*
Issues: A short description of the issues the key practice addresses according to [35].
PSE Support: Abbreviations of the PSE functions which - in our point of view - are intended to support the key practice.
Discussion: A few words of discussion on the support of the key practice by the functions named under PSE Support.

4.1 Commitment to Perform

Key Practice: *The project follows a written organizational policy for managing the system requirements allocated to software*
Issues: Three things are typically specified by the policy. First, the allocated requirements have to be documented. Second, allocated requirements have to be reviewed. The reviews may be carried out by software managers, software engineering group, software quality assurance group and others. Third, the policy specifies that changes of the allocated requirements have to cause changes of software plans, work products, and activities.
PSE Support: PMod, PMAn
Discussion: A PSE may help to ensure that requirements management follows a written policy by providing modeling functionality. The policy can explicitly be defined and represented as a process model. This facilitates understanding of the policy and

enforcement during process enaction. Process model analysis functionality is helpful in analysis of the policy in advance and determination of the effects of following different branches of the policy.

4.2 Ability to Perform

Key Practice: *For each project, responsibility is established for analyzing the system requirements and allocating them to hardware, software, and other system components.*

Issues: The responsibilities to be established covers the management and documentation of the system requirements and their changes.

PSE Support: OMod, AMan

Discussion: A PSE has great potential to support this key practice. The organizational modeling functionality enables a user to provide the organizational knowledge to a PSE necessary to support the establishment of responsibilities. This includes the definition of roles (capabilities, rights) for each organizational unit (employee, group,...) of the software development enterprise. Based on these roles the responsibilities can be established. To accomplish this task, the agent management functionality can provide some help. As described in section two, the agent management functionality manages agent-activity associations based on process behavior. This means for example, based on the requirements engineering process model developed by the use of process modeling functionality, the agent management function helps to determine agent needs (number, qualification), to adapt this needs due to change requests, and to establish the assignment of agents to subactivities of requirements engineering.

Key Practice: *The allocated requirements are documented.*

Issues: The requirements to be documented may be nontechnical or technical in nature. Nontechnical requirements are delivery dates, milestones, products to be delivered, i.e. agreements, conditions, and contractual terms of the software development process. Technical requirements on the other hand are functions of the software to be developed, its performance or the programming language to use. Further criteria usable to validate that software products satisfy requirements has to be documented.

PSE Support: CMan

Discussion: Management of requirements documentation is a task to be supported by the configuration management functionality of a PSE. Configuration management provides a versioning facility allowing to determine different versions of the requirements documents and their actuality in the software process. Further access to documents is controlled by enforcing access rights of different agents. Beyond that, the change history of the requirements documents is stored and made accessible.

Key Practice: *Adequate resources and funding are provided for managing the allocated requirements.*

Issues: Resources comprises individuals with the capabilities necessary to manage requirements as well as tools to support requirements management activities. These are among others spreadsheet programs and word processors.

PSE Support: OMod, AMan, PExec

Discussion: The management of human resources is accomplished by OMod and AMan of our PSE reference model. Organizational models allow to get an overview of the human resources situation, their number and association to projects, while AMan allows to assign individuals with the necessary capabilities as agents to a requirements management process. The process execution support functionality on the other hand allows to integrate arbitrary applications in the process supporting environment and thus to provides a very flexible collection of tools for requirements management.

Key Practice: *Members of the software engineering group and other software-related groups are trained to perform their requirements management activities.*

Issues: Training on methods, standards, procedures to be used as well as on the application domain has to be provided.

PSE Support: -

Discussion: Training of members of the requirements management group is not supported by a PSE according to our reference model.

4.3 Activities Performed

Key Practice: *The software engineering group reviews the allocated requirements before they are incorporated into the software project.*

Discussion: The purpose of these reviews is to find incomplete and missing requirements and to determine whether or not they are feasible, clearly stated, consistent, testable. Furthermore, an agreement on the requirements is established with other groups, e.g. software engineering group, testing group, quality assurance group, on the requirements is established.

PSE Support: PExec

Discussion: The only way to support this key practice by a PSE is through the process execution support functionality. A user is notified about the need to review requirements, other users participating in the review, the procedure used to review, and other conditions like time constraints. This key practice contains negotiations and thus should be supported by CSCW tools supporting negotiations between agents. Our reference model does not contain such functionality.

Key Practice: *The software engineering group uses the allocated requirements as the basis for software plans, work products, and activities.*

Issues: This key practice implies the management and control of the allocated requirements, and their usage as a basis for the software development plan and as a basis for developing the software requirements.

PSE Support: CMan, PExec

Discussion: Managing and controlling of requirements documents is a task for the configuration management functionality. Changes are incorporated in a controlled manner and version management is provided. Process execution functionality provides agents performing the planning activities with the necessary information on which requirements to use for planing, and the software engineering group with information on which documents to use as base for the software requirements.

Key Practice: *Changes to the allocated requirements are reviewed and incorporated into the software project.*

Issues: The impact of changing requirements on agreements with external and internal individuals or groups is identified and new agreements are established. Changes to software plans, work products, and activities due to changing requirements are realized.

PSE Support: PEvol, PExec

Discussion: The process evolution functionality allows to adapt the requirements engineering process to new needs, i.e. to change process definitions and notify the runtime environment about these changes. Furthermore PExec supports the new activities to be performed by user notification and tool integration as described in more detail above.

4.4 Measurement and Analysis

Key Practice: *Measurements are made and used to determine the status of the activities for managing the allocated requirements.*

Issues: Status information to be provided includes information on the state of the requirements engineering process as well as on the state of the requirements engineering documents.

PSE Support: CMan, PMon

Discussion: The configuration management provides information on the state of the

requirements documents like the actual versions and change histories. The process monitoring function provides information on the state of the requirements engineering process. Examples are the activities performed and still to perform, time constraints, agents associated to activities.

4.5 Verifying Implementation

Key Practice: *The activities for managing the allocated requirements are reviewed with senior management on a periodic basis.*

Issues: The intent of this key practice is to provide insight into software development activities of an organization at an abstract level and at certain points in time.

PSE Support: PExec, PMon

Discussion: This key practice is supported by the process execution facility as described above. The process monitoring component may provide valuable information on the state of the projects currently carried out.

Key Practice: *The activities for managing the allocated requirements are reviewed with the project manager on both a periodic and event-driven basis.*

Issues: The intent of this key practice is the same as for the one described above. The only difference is that the project manager is not only informed on a periodic basis, but also if events enforce this information.

PSE Support: PExec, PMon

Discussion: PSE support for this key practice is the same as defined for the previous one.

Key Practice: *The software quality assurance group reviews and/or audits the activities and work products for managing the allocated requirements and reports the result.*

Issues: The purpose of this key practice is to assure certain quality standards for the requirements engineering process.

PSE Support: CMan, PExec, PMon

Discussion: Configuration management functionality provides information on the work products of requirements engineering, process monitoring shows the actual state of the process, and process execution support helps to enforce the reviews and audits by user notification of tasks to be performed for quality assurance.

5. Results

Table 1 below summarizes the results of our investigation. The entry in a certain cell states how many key practices of the common feature named in the column header are supported by the PSE functionality named in the row header.

Table 1: Results

	Commitment to Perform	Ability to Perform	Activities Performed	Measurement and Analysis	Verifying Implementation	Σ
OMod	-	2	-	-	-	2
PMod	1	-	-	-	-	1
PMAn	1	-	-	-	-	1
PInst	-	-	-	-	-	0
AMan	-	2	-	-	-	2
PExec	-	1	3	-	3	7
CMan	-	-	1	1	1	3
PMon	-	-	-	1	3	4
PEvol	-	-	1	-	-	1
Σ	2	5	5	2	7	

When we consider the single functions defined for our PSE reference model, we can draw the following picture. The most important feature of a PSE for requirements management is the process execution support enforcing activities to be performed and enabling a flexible tool integration into the environment. PExec assures that requirements management is performed according to a written policy in a flexible manner. Process monitoring is also a very important PSE function. This is due to the nature of requirements management being a control process. Monitoring the actual state is necessary to achieve an efficient control and to determine problems. The importance of configuration management is based on the data intensive character of requirements engineering. Information has to be stored, managed, and access to data has to be controlled. This is especially important because a group of people participating in requirements management as well as later process phases has to use these data. Organizational modeling and agent management provides the necessary organizational knowledge needed to manage human responsibilities for requirements management and thus play an important part in supporting requirements engineering. Process modeling and process model analysis have great potential to contribute to the development of a written policy for requirements management. While process modeling supports understanding of and communication about the requirements management process, process model analysis helps to anticipate the behavior of the process when following a certain policy. Process evolution comes into play when problems in requirements management causes deviations form the initially defined process and thus is very important for trouble shouting. On the first sight, process instantiation seems to be unnecessary for PSE-based requirements management. However, although not directly supporting any key practice, PInst represents a prerequisite to enable PSE support for requirements management. The instantiation of a process is necessary to make them enactable by a PSE and thus to provide the functionality of runtime support as defined in section two. Our resume is that the functions of a PSE which are currently area of research and (partly) implemented in process supporting environments can each make a large contribution to the performance of requirements management.

Considering the results from the common features point of view we can draw the following conclusions. The commitment to perform and ability to perform common features comprises key practices to prepare the requirements management process for enactment. Actions to ensure process establishment as well as to assure preconditions have to be performed. With OMod, PMod, PMAn the modeling functionality of a PSE intended to prepare processes for performance can make valuable contributions to the associated key practices. Further actions to prepare performance but caused by dynamic process behavior are supported by PExec and AMan. The procedures necessary to implement the requirements management key process area (comprised by the activities performed common feature) are supported by PExec and CMan to enforce the data intensive requirements management process as well as by PEvol to adapt the process to new situations and conditions occurring during process enactment. The intent of the key practices associated to the measurement and analysis and verifying implementation common features is to measure and analyze the requirements management process and to induce steps to assure that the activities are performed in accordance with the defined process. Process monitoring and configuration management provides features to measure process performance and the state of the requirements documents respectively, while PExec enforces and supports performance of associated activities. In a single statement, eleven of the twelve steps of our requirements management scenario based on the CMM key process area can be supported by a PSE.

6. Conclusions

In this paper we demonstrated the feasibility of supporting requirements management by means of a certain type of CSCW applications, namely process supporting environments

(PSEs). In a nutshell, we indicated that first, each function of a generic PSE reference model based on current research contributes to requirements management and second, eleven out of twelve requirements management key practices as defined by CM/SEI's capability maturity model for software may be supported by using a PSE.
Process modeling, organizational modeling, and process model analysis as well as agent management functionality may help to prepare a requirements management process for performance. A policy is explicitly defined, its possible performance is analyzed, and organizational responsibilities for carrying out activities are established. The instantiation of the defined model makes the process controllable by the runtime support of a PSE. Process execution support and configuration management functionality helps to enforce requirements management in a flexible manner, and manages the large set of requirements documents respectively. Process monitoring and process evolution functionality supports the monitoring of the process state and the arrangement of corrective actions due to changing situations and conditions.

References

[1] The first collected Arcadia papers, UC Irvine, 1993.

[2] Armenise, P., Bandinelli, S., Ghezzi, C., Morzenti, A., "Software Process Representation Languages: Survey and Assessment", Proc. Fourth Intl. Conf. on Software Engineering and Knowledge Engineering (SEKE-4), 1992.

[3] Bandinelli, S., Fuggetta, A., Ghezzi, C., "*Software Process Model Evolution in the SPADE Environment*", IEEE Transactions on Software Engineering, Special Issue on Process Evolution, Vol. 19, No. 12, December 1993.

[4] Bandinelli, S., Fuggetta, A., Ghezzi, C., Lavazza, L., "*SPADE: An Environment for Software Process Analysis, Design, and Enactment*", in: [17], pp. 223-247.

[5] Bandinelli, S., Di Nitto, E., Fuggetta, A., "Policies and Mechanisms to Support Process Evolution in PSEEs", Proc. Third Int. Conf. on the Software Process (ICSP-3), 1994, IEEE CS Press.

[6] Belkhatir, N., Estublier, J., Melo, W., "*ADELE-TEMPO: An Environment to Support Process Modelling and Enaction*", in: [17], pp. 187-222.

[7] Canals, G., Boudjlida, N., Derniame, J.-C., Godart, C., Lonchamp, J., "*ALF: A Framework for Building Process-Centered Software Engineering Environments*", in: [17], pp. 153-185.

[8] Conradi, R., Fernström, C., Fuggetta, A., Snowdon, R., "*Towards a Reference Framework for Process Concepts*", Proceedings 2nd European Workshop on Software Process Technology (EWSPT-2), Trondheim, Norway, Sep. 1992, LNCS 635, Springer Verlag, pp. 3-17.

[9] Conradi, R., "*Resume for Session on Process Change at ISPW-8*", Proceedings 8th Intl. Software Process Workshop (ISPW-8), Wadern, Germany, March 1993, IEEE CS Press, pp. 15-17.

[10] Conradi, R., Fernström, C., Fuggetta, A., "*Concepts for Evolving Software Processes*", in: [17], pp. 9-31.

[11] Curtis, B., Kellner, M. I., Over J., "*Process Modeling*", Communications of the ACM, Vol. 35, No. 9, September 1992, pp. 75-90.

[12] Dart, S. A., "*The Past, Present, and Future of Configuration Management*", CMU/SEI-92-TR-8, July 1992, 28 p.

[13] Deiters, W., Gruhn, V., "*Managing Software Processes in the Environment MELMAC*, Proceedings of the Symposium on Practical Software Development Environments, ACM, 1990, pp. 193-205.

[14] Ellis, C. A., Gibbs, S. J., Rein, G. L., "*Groupware - Some Issues and Experiences*", Communications of the ACM, Vol. 34, No. 1, January 1991, pp. 39-58.

[15] Feiler, P. H., "*Configuration Management Models in Commercial Environments*", CMU/SEI-91-TR-7, March 1991, 54 p.

[16] Feiler, P. H., Humphrey W. S., "*Software Process Development and Enactment: Concepts and Definitions*", Proc. Second Int. Conf. on the Software Process (ICSP-2), 1993, IEEE CS Press.

[17] Finkelstein, A., Kramer, J., Nuseibeh, B., (Eds.), "*Software Process Modelling and Technology*", John Wiley & Sons, 1994, 362 p.

[18] Greif, I., (Ed), "*Computer Supported Cooperative Work: A Book of Readings*", Morgan Kaufmann Publishers, 1988.

[19] Gruhn, V., Saalmann, A., "*Software Process Validation Based on FUNSOFT Nets*", Proceed-

ings 2nd European Workshop on Software Process Technology (EWSPT-2), Trondheim, Norway, Sep. 1992, LNCS 635, Springer Verlag, pp. 223-226.
[20] Herbsleb, J., Carleton, A., Rozum, J., Siegel, J., Zubrow, D., "*Benefits of CMM-Based Software Process Improvement: Initial Results*", CMU/SEI-94-TR-13, August 1994.
[21] Huff, K. E., Kaiser, G. E., "*Change in the Software Process*", Proceedings 7th Intl. Software Process Workshop (ISPW-7), Yountville, California, October 1991, IEEE CS Press, pp. 10-13.
[22] Huff, K. E., "*Software Process Measurement Session Summary*", Proceedings 8th Intl. Software Process Workshop (ISPW-8), Wadern, Germany, March 1993, IEEE CS Press, pp. 18-21.
[23] Humphrey, W. S., "*Characterizing the Software Process: A Maturity Framework*", IEEE Software, Vol. 5, No. 2, March 1988, pp. 73-79.
[24] Humphrey, W. S., Snyder, T. R., Willis, R. R., "*Software Process Improvement at Hughes Aircraft*", IEEE Software, July 1991, pp. 11-23.
[25] Jaccheri, M. L., Conradi, R., "*Techniques for Process Model Evolution in EPOS*", IEEE Transactions on Software Engineering, Special Issue on Process Evolution, Vol. 19, No. 12, December 1993, pp. 1145-1157.
[26] Jamart, P., van Lamswerde, A., "*A Reflective Approach to Process Model Customization, Enactment, and Evolution*", Proc. Third Int. Conf. on the Software Process (ICSP-3), 1994, IEEE CS Press.
[27] Johansen, R., "*Groupware: Computer Support for Business Teams*", The Free Press, New York, 1988.
[28] Kaiser, G. E., Ben-Shaul, I. Z., "*Process Evolution in the Marvel Environment*", Proceedings 8th Intl. Software Process Workshop (ISPW-8), Wadern, Germany, March 1993, IEEE CS Press, pp. 104-106.
[29] Kellner, M. I., Hansen, G. A., "*Software Process Modeling: A Case Study*", Proceedings 22nd Intl. Hawaii Conf. on System Science, Hawaii 1989, IEEE CS Press, pp.175-188.
[30] Kitson, D. H., Masters, S. M., "*An Analysis of SEI Software Process Assessment Results: 1987-1991*", Proc. 15th Intl. Conf. on Software Engineering, 1993, IEEE CS Press, pp. 68-77.
[31] Lonchamp, J., "*An Assessment Exercise*", in: [17], pp. 335-356.
[32] Madhavji, N. H., Hoeltje, D., Hong, W. K., Bruckhaus, T., "*Elicit: A Method for Eliciting Process Models*", Proc. Third Int. Conf. on the Software Process (ICSP-3), 1994, IEEE CS Press.
[33] Oquendo, F., "*SCALE: Process-centered Environments for System Composition Reuse*", Proceedings 2nd European Workshop on Software Process Technology (EWSPT-2), Trondheim, Norway, Sep. 1992, LNCS 635, Springer Verlag, pp. 403-412.
[34] Paulk, M. C., Curtis, B., Chrissis, M. B., Weber C. V., "*Capability Maturity Model for Software, Version 1.1*", CMU/SEI-93-TR-24, February 1993.
[35] Paulk, M. C., Weber, C. V., Garcia, S., Chrissis, M. B., Bush, M., "*Key Practices of the Capability Maturity Model, Version 1.1*", CMU/SEI-93-TR-25, February 1993.
[36] Pohl, K., "*The Three Dimensions of Requirements Engineering: A Framework and its Applicaitons*", Information Systems, Vol. 19, No. 3, 1994, pp. 243-258.
[37] Prinz, W., "*TOSCA - Providing organizational information to CSCW applications*", Proc. 3rd European Conf. on Computer Supported Cooperative Work (ECSCW'93), Kluwer, 1993, pp. 139-154.
[38] Prinz, W., "*Object-Oriented Organization Modeling for the Support of CSCW*", Proc. 27th Hawaii International Conf. on System Sciences, Vol. IV: Information Systems, IEEE CS Press, 1994, pp. 797-806.
[39] Royce, W. W., "*Managing the Development of Large Software Systems*", IEEE WESCON, August 1970, reprinted in: Proc. 9th Intl. Conf. on Software Engineering, IEEE CS Press, 1987, pp. 328-338.
[40] Rupietta W., "*Organization Models for Cooperative Office Applications*", Proc. 5th Int. Conf. on Database and Expert Systems Applications (DEXA), Athens, Greece, September 1994.
[41] Thomas, I., Nejmeh, B. A., "*Definitions of Tool Integration for Environments*", IEEE Software, March 1992, pp. 29-35.